CS 2150 Exam 2, spring 2014

Name _____

You MUST write your e-mail ID on **EACH** page and bubble in your userid at the bottom of this first page. And put your name on the top of this page, too.

If you are still writing when "pens down" is called, your exam will be ripped up and not graded – even if you are still writing to fill in the bubble form. So please do that first. Sorry to have to be strict on this!

Other than bubbling in your userid at the bottom of this page, please do not write in the footer section of this page.

There are 8 pages to this exam. Once the exam starts, please make sure you have all the pages. Questions are worth different amounts of points.

If you do not bubble in this first page properly, you will not receive credit for the exam!

Answers for the short-answer questions should not exceed about 20 words; if your answer is too long (say, more than 30 words), you will get a zero for that question!

This exam is CLOSED text book, closed-notes, closed-calculator, closed-cell phone, closed-computer, closed-neighbor, etc. Questions are worth different amounts, so be sure to look over all the questions and plan your time accordingly. Please sign the honor pledge below.

_____

_____

_____

_____

Three things are certain:
Death, taxes, and lost data.
Guess which has occurred.

(the bubble footer is automatically inserted into this space)

Page 2: C++ and big-Oh

1. [3 points] Why does C++ treat array base names as *constant* pointers?

2. [6 points] Prove that $5n^2 \in O(n^3)$. Show your work.

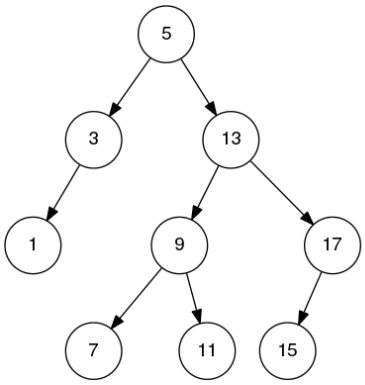
3. [3 points] References were created to be a "safe" type of pointer, meaning that certain invalid operations that one could do with pointers (most of which would cause a segemntation fault) were not allowed with references. What are three such invalid operations?

Page 3: Trees

4. [4 points] Describe the binary search tree removal algorithm.

5. [5 points] What are the properties of a red-black tree?

6. [3 points] Insert 6 into the AVL tree shown below. Show the resutling tree.



Page 4: Hashes

7. [8 points] Consider the four collision resolution strategies we have discussed in class. Assume we have a hash table of size 10, and our primary hash function for integers is $h(x) = x \bmod 10$. A secondary hash function, if needed, is $h_2(x) = (x \bmod 5) + 1$. Your task is to generate, for each of the four strategies, a sequence of three numbers to insert such that the second inserted number collides with the first (and thus probes (or chains) to a new spot), and the third inserted number collides with the second (and thus probes (or chains) to a new spot). Each strategy will likely have a different sequence of three numbers. Your numbers should be properly inserted into the hash table diagrams below, as per the appropriate collision resolution strategy. List the numbers in their insertion order underneath the table (i.e., if your numbers are 1, 2, and 3, then you should list "1,2,3" under that table).

	Linear probing	Quadriatic probing	Double hashing	Separate chaining
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				

8. [4 points] Which is more likely to occur, the linear run-time of a binary search tree, or the linear run-time of a hash table? Why?

Page 5: IBCM

9. [9 points] You are to implement, in IBCM, the equivalent of the C++ statement: `x = a[i];`. Assume that `x` (where to store the value), `a` (the array base address), and `i` (the index into the array) are currently defined variables somewhere in memory. Your answer should be in IBCM opcodes (i.e., `add foo`), and ***not*** hexadecimal code. You may define other values and labels as necessary, but clearly indicate what those values are. Your answer should be an IBCM code snippet (enough to accomplish this task), and should ***not*** be a complete IBCM program. The listing of IBCM opcodes is to the right, for your reference.

Hex	Opcode
0	halt
1	I/O
2	shifts
3	load
4	store
5	add
6	sub
7	and
8	or
9	xor
a	not
b	nop
c	jmp
d	jmpe
e	jmpl
f	brl

10. [3 points] Why did we bother learning IBCM in this course?

Page 6: x86 (and a bit of C++)

11. [3 points] Name two changes that one could make that would help prevent buffer overflow attacks. These changes could be to the C++ language, to the compiler, or to the C calling convention, but they must be realistic (i.e., you can't suddenly add bounds checking without completely redesigning the C++ language).

12. [3 points] When might you *not* push `ebp` onto the stack as part of the activation record?

13. [3 points] List 3 things that a compiler might do when the code is compiled with an optomization flag (such as `-O2`). In other words, describe 3 different types of optimizations that one can make in x86 assembly.

14. [3 points] List 6 C++ flags, and *briefly* explain what they do.

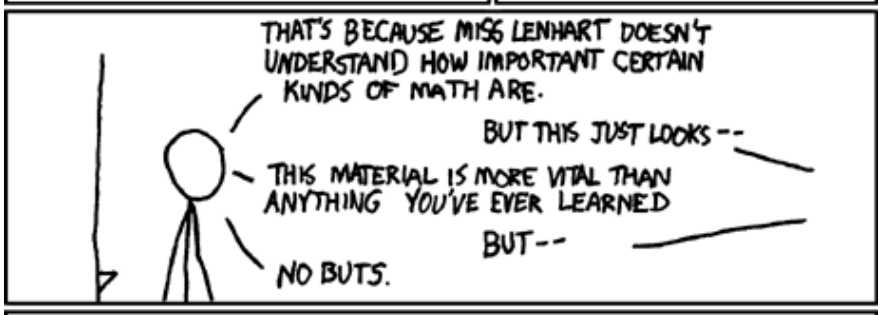
Page 7: Miscellaneous

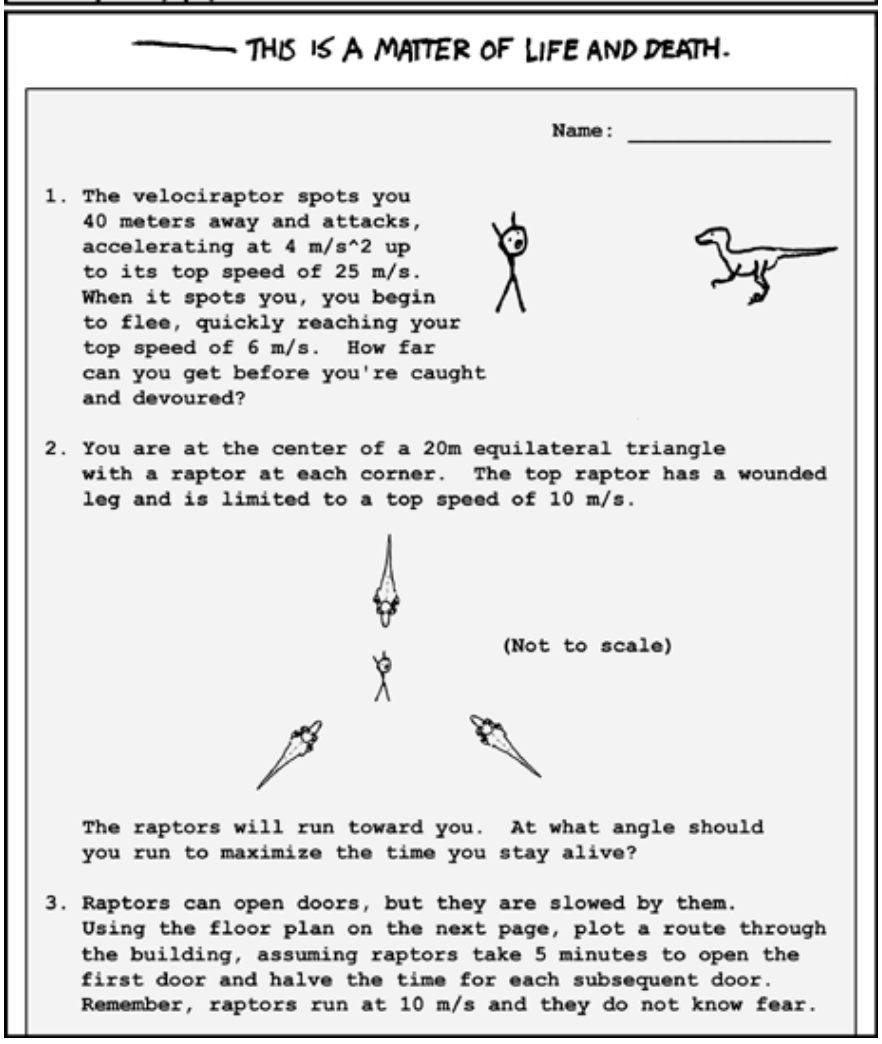
15. [3 points] Briefly define the following terms, in the context of makefiles: target, dependency, suffix rule.

16. [3 points] How does an algorithm (method, etc.) know which specific type of rotation to perform to properly balance an unbalanced AVL tree?

17. [6 points] What is the maximum (finite) value that can be held in a `double`? You may leave it as a formula. Show your work!

Page 8: Comic!





https://xkcd.com/135/